Point-to-CAD 3D Registration Algorithm for Relative Navigation Using Depth-Based Maps

Antonio Terán Espinoza
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
teran@mit.edu

Timothy P. Setterfield
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Dr.
Pasadena, CA 91109
timothy.p.setterfield@jpl.nasa.gov

***Abstract*— This paper presents an end-to-end 3D registration algorithm for relative navigation between known objects based on a point-to-CAD iterative closest point (ICP) principle. The objective of this method is to take in a measured point cloud extracted from a depth or disparity map – such as the ones obtained from stereo cameras, time-of-flight cameras, LiDARs, or depth from defocus sensors – and calculate the rigid body transformation that best aligns the measured data with a corresponding 3D CAD model. By leveraging the geometric information encoded into stereolithography (STL) files, it is sought to address the computational intractability imposed by the naïve generation of dense target point clouds solely based on the target's known surface. To this end, the proposed approach computes a bijective projection onto the known triangular mesh to obtain a target point cloud with which to use ICP techniques for incremental alignment; the projection step is then carried on recursively until the convergence criteria are met, yielding a relative 6DOF pose between the two objects to be used within the estimation pipeline. Demonstrations of the algorithm are presented using simulated datasets; results include time complexity analyses for real-time operation cases, performance variation assessments with respect to CAD model complexity, and sensitivity analysis for determining the tolerance to distinct noise levels and spurious measurements. The design and implementation of the algorithm makes use of the open-source Point Cloud Library, and access to its source code is included within this work.**

TABLE OF CONTENTS

1. INTRODUCTION

Close proximity operations and relative navigation maneuvers are an essential component of future space missions, including on-orbit servicing, in-space robotic assembly, and orbital debris removal [1], [2]. To successfully and autonomously carry out such tasks, relative state estimation between all operating agents is paramount. Within this estimation pipeline, depth sensing becomes a crucial ability regarding the acquisition of information with which inference is to be performed [3]. Thus, given the increasing interest and use of geometric representations for relative estimation techniques – be it through active methods such as LiDAR [4], or passive approaches such as stereo vision [5] – this work presents an end-to-end 3D registration algorithm for relative navigation between known objects, based on a point-to-CAD iterative closest point (ICP) principle.

Geometric registration is defined as the act of associating distinct sets of collected data into one common frame of reference by minimizing the alignment error between pairs of such sets [6]. The sets of data being cosidered by this work involve the three-dimensional depth information of the observed object acquired by the inspecting satellite, e.g., the sets of point clouds being acquired by the spacecraft's sensor suite at each time step.

The scenarios herein considered involve an inspector satellite equipped with a 3D sensor – such as a stereo camera, time-of-flight camera, etc. – and a target object with a known three-dimensional structure. The output of the 3D sensor is taken to be a depth-based map, such as a disparity map or a depth map, which can then be used to reproject a point cloud into 3D space. With respect to the *a priori* structure information, it is likely for this to be easily obtainable due to the meticulous nature of space operations [7], in which spacecraft designs are thoroughly analyzed and reviewed, and from which a coarse CAD model can provide the warranted information. Depending on the scenario to be considered, the inspected object can represent either active or passive spacecraft to be serviced, a component of an in-space assembly process, or space debris to be deorbited.

2. BACKGROUND AND RELATED WORK

The use of registration algorithms within the robotics field was originally proposed a couple of decades ago, mainly to address visual navigation and robot vision problems [8]. To date, the objective and problem formulation of such algorithms remain, at a high level, mostly unchanged.

Taking into account a source and a target point cloud, the objective consists in finding the transformation $\mathbf{T}$ between the two that minimizes the distance between pairs of points. In this case, point clouds correspond to sets of geometric data of the form

$$ {}^{A}\mathcal{X} = \left\{ \mathbf{x}_i : \mathbf{x}_i \in \mathbb{R}^3 \ \forall \ i \in [1, \cdots, N] \right\}, \tag{1} $$

where N indicates the number of set elements, or points, in the point cloud, whose 3D coordinates are expressed with respect to coordinate frame A. Thus, with ${}^{A}\mathcal{X}$ and ${}^{B}\mathcal{Y}$ as target and source point clouds, the problem formulation can

978-1-5386-6854-2/19/$31.00 ©2018. California Institute of Technology. Government sponsorship acknowledged.

be expressed as finding

$$ {}^{A}_{B}\mathbf{T}^{\star} = \arg\min_{\mathbf{T}} \; \mathbf{d}\left({}^{A}_{B}\mathbf{T}({}^{B}\mathcal{Y}), {}^{A}\mathcal{X}\right), \tag{2} $$

where the sought transformation is of the form ${}^{A}_{B}\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0^T} & 1 \end{bmatrix}$, with $\mathbf{R} \in \mathrm{SO}(3)$ and $\mathbf{t} \in \mathbb{R}^3$ representing the relative rotation and translation from frame B to frame A, and the function $\mathbf{d}(\cdot,\cdot)$ denotes the alignment error between point clouds as $\mathbf{d}\left({}^{A}\mathcal{Y}, {}^{A}\mathcal{X}\right) = \sum_{i=1}^{N} \| \mathbf{y}_i - \mathbf{x}_i \|_2$.

The most common algorithm family employed to solve such problems is that of Iterative Closest Point (ICP) methods [9]. For problems fitting the aforementioned formulation, i.e., those in which the correspondence between points in both sets is available, the point-to-point ICP algorithm can be employed [10]. Nevertheless, depending on the problem at hand, the target point clouds cannot always be represented as sets of 3D points, therefore, multiple types of ICP variants have arisen: point-to-line ICP [11], point-to-surface ICP [12], point-to-curve ICP, amongst others [6].

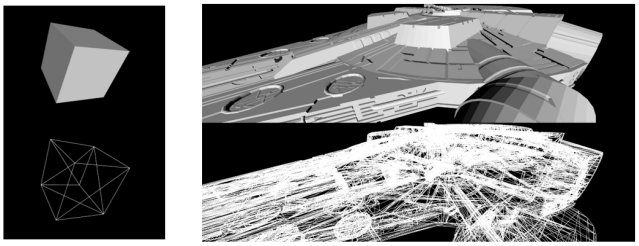
In the scenarios herein considered, after acquiring a sample point cloud of the observed object, of which a CAD model (known 3D structure) is available, an ICP variant of the type point-to-CAD is warranted for addressing the registration problem, since no specific target point cloud is initially available. Previous work pertaining such a technique has been carried out, in which point clouds obtained from LiDAR scans are aligned to an initial target point cloud obtained from a CAD model, with the aid of additional sensor information (IMU data) and a Kalman filter [13], [14]. Nonetheless, no specific methodologies for constructing a target point cloud from CAD models are discussed.

This aformentioned issue is considered in [15], and a methodology based on STL (STereo Lithography) CAD models is proposed. The work here presented builds on top of this strategy, adapting it to account for the nature of working with depth-based reprojected point clouds, whose size and shape tend to be quite irregular due to the fact that they are highly depdendent on the quality of the scene and the vantage point of the inspector satllite; that is, whenever an object is inspected from afar, a small point cloud is obtained, whereas when the observed object encompasses a sizeable portion of the inspector's field of view, the obtained point cloud is much larger in size.

Similarily, the proposed strategy makes use of STL files and its encoded geometric information in order to address the computational intractability imposed by the naïve generation of dense target point clouds solely based on the target's known surface, as proposed in [15]. Stereolithography files describe the surface geometry of a 3D object $\mathbf{\Omega}$ as a set $\{\mathbf{\Upsilon}, \mathbf{\Phi}\}$, in which $\mathbf{\Upsilon}$ represents the set of all "N" surface vertices $\boldsymbol{\upsilon}_i \in \mathbf{\Upsilon}$, and $\mathbf{\Phi}$ the mesh of all "M" triangular polygons $\boldsymbol{\phi}_j \in \mathbf{\Phi}$, with $i \in [1,\ldots,N]$ and $j \in [1,\ldots,M]$ [16]. Two such sample files are shown in Figure 1.

3. Problem Formulation

A depiction of the scenario to be considered is shown in Fig. 2, in which an inspector satellite – assumed static and positioned at the origin of the world coordinate frame W – passively inpsects a resident space object. At each time step, the 3D depth sensor on-board the inspector satellite outputs a depth-based map $\mathbf{M} \in \mathbb{R}^{p \times q}$. This depth-based map is represented as a two-dimensional matrix whose $\mathbf{m_{ij}} \in \mathbb{R}^{+}$ value represents the depth at that corresponding location/pixel, $\forall i \in [1,\ldots,p]$ and $\forall j \in [1,\ldots,q]$.



(a) simple STL (b) complex STL

Figure 1. An STL file of a cube (a) with $N = 8$ vertices and $M = 12$ triangular polygons, with dimensions of $1m \times 1m \times 1m = 1m^3$. A more complex STL model is shown in (b), with $N = 66,201$ vertices and $M = 149,726$ triangular polygons.

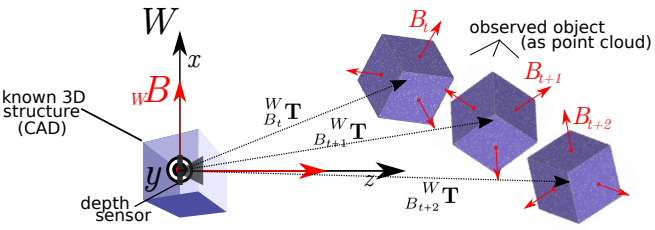


Figure 2. Pictorial depiction of the problem formulation. As shown in the image, the depth sensor's coordinate frame is aligned with the world coordinate frame W; additionally, the body frame B of the projected CAD model, which is the object with which the registration takes place, is also aligned with the coordinate frame W. At time step t, the inspector satellite observes the target object and its body frame B_t as a point cloud ${}^{B_t}\mathcal{X}$. The objective is then to estimate the relative transformation ${}^{W}_{B_t}\mathbf{T}$ between frames B_t and W.

By making use of the point cloud ${}^{B_t}\mathcal{X}$ computed from its corresponding depth map $\mathbf{M}_t$, the underlying problem then becomes the calculation of the 6DOF relative transformation ${}^{W}_{B_t}\mathcal{T}$ between the world frame and the target object's current frame. By doing so, a pose estimate measurement for feeding into the inspector's estimation pipeline is able to be obtained, enabling proximity operation maneuvers between the two objects.

4. Approach

The structure of the proposed algorithm to compute the relative transformation between the two objects is composed of five main steps: i) preprocessing step, in which filters are used to add information to the 3D object $\mathbf{\Omega}$; ii) reprojection step, in which the initial 3D source point cloud is computed; iii) target point cloud step, in which the CAD model is utilized to compute a target point cloud; iv) registration step, in which ICP methods are employed to incrementally align the source and target point clouds; and v) recursion, in which steps iii) and iv) are looped until the convergence criteria are met.

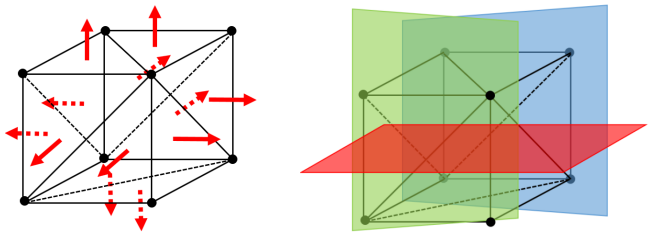
The resulting algorithm is summarized in Algorithm 1, and a detailed explanation is given in the following subsections.

Algorithm 1 Point-to-STL Registration

procedure GET6DOFPOSE($\mathbf{M}_t, \mathbf{\Omega}$)
  Precompute surface normals from $\{\boldsymbol{\Upsilon}, \boldsymbol{\Phi}\}$
  Setup k-d trees data structure
  ${}^{B_t}\mathcal{X} \leftarrow$ reproject depth-based map $\mathbf{M}_t$
  ${}^{W}_{B_t}\mathcal{T} \leftarrow \{\varnothing\}$
  $error \leftarrow \infty$
  while $error > \epsilon_d$ **do**
    ${}^{W}\mathcal{X} \leftarrow \{\varnothing\}$
    for each $\mathbf{x}_i \in {}^{B_t}\mathcal{X}$ **do**
      ${}_{k}\mathbf{Ne}_{\boldsymbol{\Upsilon}} \leftarrow$ find $\mathbf{x}_i$ neighbors
      $\boldsymbol{\Phi}' \leftarrow$ get interesting polygons from ${}_{k}\mathbf{Ne}_{\boldsymbol{\Upsilon}}$
      $\boldsymbol{\Pi}_{\boldsymbol{\Phi}'} \leftarrow \{\varnothing\}$
      for each $\phi_z \in \boldsymbol{\Phi}'$ **do**
        $\boldsymbol{\pi}_{\boldsymbol{\Phi}} \leftarrow$ projection of $\mathbf{x}_i$ onto ϕ_z
        $\boldsymbol{\Pi}_{\boldsymbol{\Phi}'} \leftarrow \boldsymbol{\Pi}_{\boldsymbol{\Phi}'} \cup \boldsymbol{\pi}_{\boldsymbol{\Phi}}$
      $\boldsymbol{\pi}^*_{\boldsymbol{\Phi}} \leftarrow \min_{\boldsymbol{\pi}} \boldsymbol{\Pi}_{\boldsymbol{\Phi}'}$
      ${}^{W}\mathcal{X} \leftarrow {}^{W}\mathcal{X} \cup \boldsymbol{\pi}^*_{\boldsymbol{\Phi}}$
    ${}^{W}_{B_t}\mathbf{T}, {}^{B_t}\mathcal{X} \leftarrow \text{ICP}({}^{W}\mathcal{X}, {}^{B_t}\mathcal{X})$
    ${}^{W}_{B_t}\mathcal{T} \leftarrow {}^{W}_{B_t}\mathcal{T} \cup {}^{W}_{B_t}\mathbf{T}$
    $error \leftarrow \frac{1}{p}\mathbf{d}\left({}^{W}_{B_t}\mathbf{T}({}^{B_t}\mathcal{X}), {}^{W}\mathcal{X}\right)$
  ${}^{W}_{B_t}\hat{\mathbf{T}}^* \leftarrow \prod_{i}{}_{i-1}{}^{i}\mathbf{T}, \forall \mathbf{T} \in {}^{W}_{B_t}\mathcal{T}$
  return ${}^{W}_{B_t}\hat{\mathbf{T}}^*$

i) Preprocessing

The preprocessing is to be performed only once for each employed CAD model, and consists in precomputing the normal vector corresponding to each triangular mesh (Fig. 3a), as well as in arranging the model's vertices into a k-d tree data structure, which allows for efficient spatial searches along the vertex set $\boldsymbol{\Upsilon}$ of the STL shape (Fig. 3b).



(a) surface normals **(b) geometric data structure**

Figure 3. Preprocessing step depicting the two operations to be carried out upon each utilized CAD model: (a) normal vector calculation for each triangular polygon, and (b) a k-d tree data structure for efficient vertex search.

ii) Map Reprojection Step

In order to obtain the initial source point cloud at each time step, the depth-based map values need to be projected into 3D space to obtain ${}^{B_t}\mathcal{X}$; for camera-based sensor information, the geometry of similar triangles and a pinhole camera model can be employed to obtain the following set of equations

$$X_{ij} = (i - p_i) \cdot \frac{\mathbf{m_{ij}}}{f},\ Y_{ij} = (j - p_j) \cdot \frac{\mathbf{m_{ij}}}{f},\ Z_{ij} = \mathbf{m_{ij}} \quad (3)$$

for a depth map $\mathbf{M}_t$, where the reprojected point $\mathbf{x}_k = (X_{ij}, Y_{ij}, Z_{ij})$ is to be added to the point cloud ${}^{B_t}\mathcal{X}$, the pair of points (p_i, p_j) represent the camera's principal point coordinates, and f its focal length. A similar set of reprojection equations can be utilized whenever $\mathbf{M}_t$ denotes a disparity map obtained from stereo cameras.

iii) Compute Target Point Cloud

The objective of this step is to compute a bijective projection of the source point cloud ${}^{B_t}\mathcal{X}$ onto the known triangular mesh $\boldsymbol{\Phi}$ to obtain a target point cloud ${}^{W}\mathcal{X}$ with which to use ICP techniques for incremental alignment. In order to do so, it is necessary to calculate the best projection of $\mathbf{x}_i$ onto the CAD's mesh $\boldsymbol{\Phi}$ to obtain $\boldsymbol{\pi}^*_{\Phi}(\mathbf{x}_i)$, iterating over every $\mathbf{x}_i \in {}^{B_t}\mathcal{X}$. The proposed process for achieving this consists of the following series of steps.

Find k-nearest neighbors of $\mathbf{x}_i$ in the vertex set $\boldsymbol{\Upsilon}$ – Using the precomputed k-d tree structure, the set

$${}_{k}\mathbf{Ne}_{\boldsymbol{\Upsilon}}(\mathbf{x}_i) = \{\boldsymbol{\upsilon}_j : \mathbf{d}(\boldsymbol{\upsilon}_j, \mathbf{x}_i) \leq \mathbf{d}(\boldsymbol{\upsilon}_q, \mathbf{x}_i), \forall \boldsymbol{\upsilon}_q \in \boldsymbol{\Upsilon}\},$$

can be efficiently computed, for $i \in [1, \ldots, k], q \neq i$.

Retrieve the interesting polygons – A subset $\boldsymbol{\Phi}' \subset \boldsymbol{\Phi}$ of polygons that contain as a vertex at least one of the k-nearest neighbors of $\mathbf{x}_i$ is to be calculated. That is, using the precomputed data structure for fast access, getting the set

$$\boldsymbol{\Phi}' = \{\boldsymbol{\phi}_z = (\boldsymbol{\upsilon}_i, \boldsymbol{\upsilon}_j, \boldsymbol{\upsilon}_l) \in \boldsymbol{\Phi} : \boldsymbol{\upsilon}_\cdot \in {}_{k}\mathbf{Ne}_{\boldsymbol{\Upsilon}}(\mathbf{x}_i), \forall \boldsymbol{\phi}_z \in \boldsymbol{\Phi}\}.$$

Compute projection of $\mathbf{x}_i$ onto each polygon $\boldsymbol{\phi}_z \in \boldsymbol{\Phi}'$ – First, by making use of the precomputed surface normals, a projection onto the plane defined by $\boldsymbol{\phi}_z$ is calculated. By means of barycentric coordinates, the projection is to be ensured to lie within the triangular mesh. Subsequently, the projected point $\boldsymbol{\pi}_{\boldsymbol{\phi}_z}(\mathbf{x}_i)$ is added onto the projection set $\boldsymbol{\Pi}_{\boldsymbol{\phi}_z}$.

Choose best projection – From the projection set $\boldsymbol{\Pi}_{\boldsymbol{\Phi}'}$, the best projected point $\boldsymbol{\pi}^*_{\Phi}(\mathbf{x}_i)$, i.e., the one with the shortest Euclidean distance with respect to $\mathbf{x}_i$, is chosen and added onto the the target point cloud ${}^{W}\mathcal{X}$.

iv) Registration Step

After iterating over every element in the source point cloud, the bijective projection ${}^{W}\mathcal{X}$ is set as the ICP's target point cloud. Given the fact that no correspondance problem between the two point clouds is present, traditional point-to-point Iterative Closest Point methods are employed for incremental alignment [17]. After running the ICP methods, a transformation estimate ${}^{W}_{B_t}\hat{\mathbf{T}}$ between the two point clouds is obtained.

v) Recursion

As it can be anticipated, the target point cloud ${}^{W}\mathcal{X}$ built in step *iii)* is not necessarily the approriate target point cloud for being able to fully register the initial source point cloud and the CAD model's surface. This problem is shown in Figure 4b), in which the computed target point cloud can be seen only spanning one side of the model's surface.

To overcome this issue, a recursion over steps *iii)* and *iv)* is to be carried out. By doing so, once the point-to-point ICP

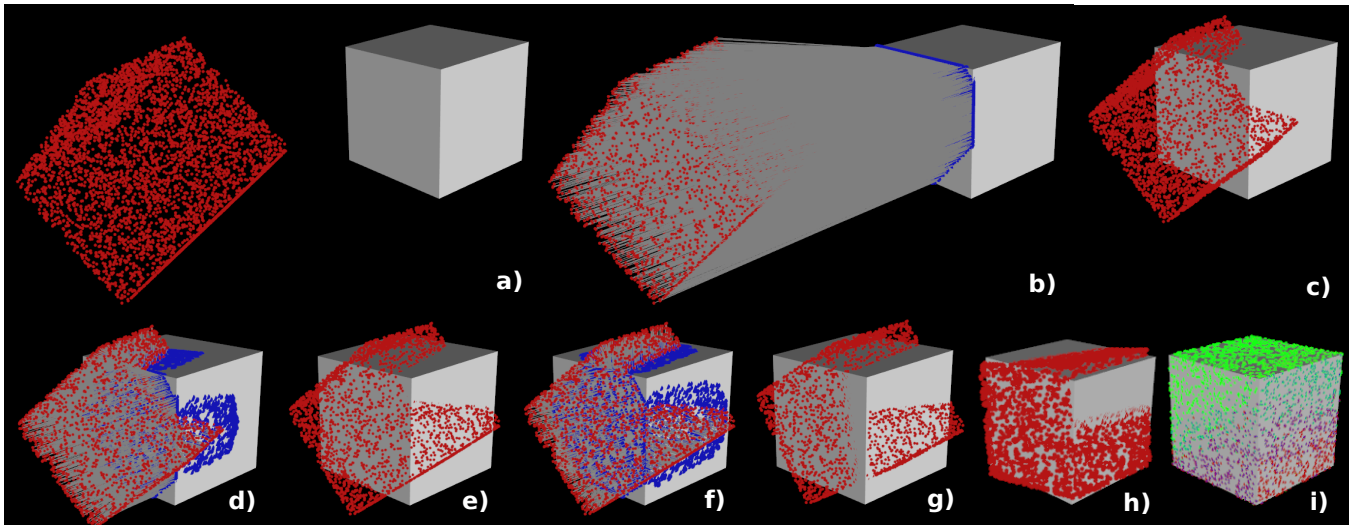


Figure 4. Registration sequence example: a) initial conditions are shown, with the source point cloud ${}^{B_t}\mathcal{X}_0$ shown in red; b) the computation of the target point cloud ${}^{W}\mathcal{X}_0$ (blue) is showcased, in which the bijective projection onto the CAD's surface, for each $\mathbf{x}_i \in {}^{B_t}\mathcal{X}_0$, is denoted by a projection line; c) resulting source point cloud ${}^{B_t}\mathcal{X}_1$ after running ICP$({}^{W}\mathcal{X}_0, {}^{B_t}\mathcal{X}_0)$; d) start of the second iteration of the algorithm, in which ${}^{B_t}\mathcal{X}_1$ (red) is used to project the second target point cloud ${}^{W}\mathcal{X}_1$ onto the mesh Φ; e) registration via ICP$({}^{W}\mathcal{X}_1, {}^{B_t}\mathcal{X}_1)$, resulting in ${}^{B_t}\mathcal{X}_2$ (red); f) projection step for obtaining ${}^{W}\mathcal{X}_2$; g) ICP step resulting in ${}^{B_t}\mathcal{X}_3$; h) subsequent source point cloud ${}^{B_t}\mathcal{X}_4$; i) final result, in which the projected cloud (blue) and the source cloud (red) are entirely overlapped with the ground truth values (shown in green).

in the registration step arrives at a local minimum, a new target point cloud ${}^{W}\mathcal{X}_1$ is computed by taking into account the modified position of the source point cloud. This newly computed target point cloud is then passed to the ICP method and is set as a new objective. This is repeated as many times as desired, or until reaching certain stoppage criteria.

Algorithm convergence is assummed any time the value obtained after computing the error between the source and target point clouds using the current transformation estimate ${}^{W}_{B_t}\hat{\mathbf{T}}$ falls below a chosen threshold ϵ_d, i.e.,

$$\mathbf{d}({}^{W}_{B_t}\hat{\mathbf{T}}({}^{B_t}\mathcal{X}), {}^{W}\mathcal{X}) \leq \epsilon_d.$$

Additionally, stoppage conditions are put in place to identify problems such as incorrect convergence to local minima by assessing the Frobenius norm of the difference between two subsequent transformations.

5. Results and Discussion

Implementation

The proposed algorithm was implemented in C++ using the open-source Point Cloud Library (PCL) [18], mostly for its visualization support and its built-in Iterative Closest Point methods. All images were produced via the VTK library, and the employed depth and disparity-based maps, along with its corresponding point clouds, were produced with the help of Blender [19], an open-source 3D computer graphics software. All tests herein reported were conducted using an Intel Xeon E3-1505M 2.8 GHz laptop with 64 GB of available RAM.

Test Cases

To showcase the results for each step, a sample registration procedure using a small CAD model is presented. Using the STL file shown in Figure 1a, an initial and noiseless source point cloud with 5,000 elements is sampled along its surface, and a random rigid body transformation – which acts as a ground truth for the experiment – is applied to all points. The sequence of steps is depicted in Figure 4.

In order to assess the performance of the registration algorithm's final pose estimate, the average point disparity between two clouds is calculated as

$$\eta = \frac{1}{p} \cdot \mathbf{d}\left({}^{W}_{B_t}\hat{\mathbf{T}}^{*}({}^{B_t}\mathcal{X}_0), {}^{W}\mathcal{X}_{\mathbf{G}}\right), \tag{4}$$

where p denotes the number of points in the point cloud, and ${}^{W}\mathcal{X}_{\mathbf{G}}$ represents the ground truth point cloud obtained by sampling the surface of the CAD model. For the particular set of examples similar to the one shown in Figure 4, a desired accuracy of $\eta_d = 1\mathrm{e}^{-5}\ m/\mathtt{point}$ was chosen. Such threshold was reached at an average of **19 iterations**, with an average **runtime of 30.28 seconds**.

A more challenging registration example is shown in Figure 5, in which a CAD model with $N = 2,689$ vertices and a triangular mesh with $M = 5,374$ polygons is employed. An initial and noiseless ground truth point cloud ${}^{W}\mathcal{X}_{\mathbf{G}}$ of 5,000 elements is sampled from its surface, and a random rigid body transformation is applied in order to set the registration algorithm's initial conditions (resulting in the black point cloud shown in Fig. 5a). The same desired accuracy ($\eta_d = 1\mathrm{e}^{-5}\ m/\mathtt{point}$) and stopping conditions used for the previous example are chosen. It's worth noting that even though the same level of accuracy was able to be

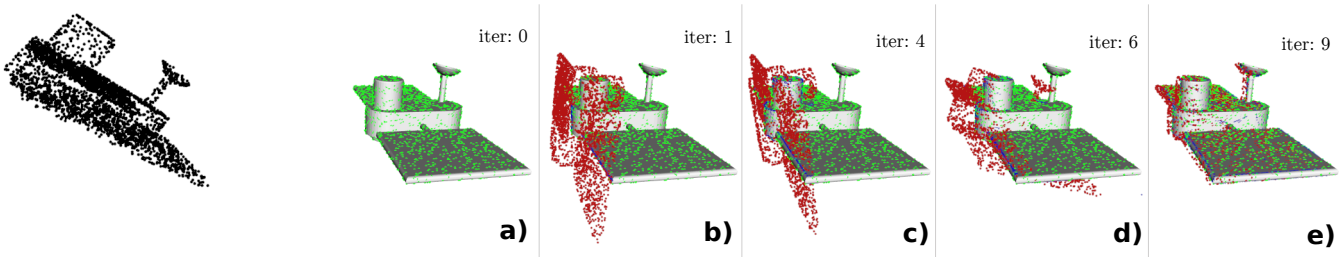


Figure 5. Registration sequence using Ω with $N = 2,689$ vertices and $M = 5,374$ polygons: a) initial conditions are shown; b) resulting alignment after initial point cloud projection and ICP step; c) the source point cloud ${}^{B_t}\mathcal{X}_4$ (shown in red) starts to progressively rotate due to the projected target point cloud ${}^{W}\mathcal{X}_4$'s alignment (blue). d) source point cloud's alignment almost complete; e) clear translation between previously shown step, in which the source point cloud starts to incrementally move towards the model's center of geometry.

obtained, the number of average iterations, as well as the average required computation time for a full registration to be performed, increased significantly: for this case, an average of **29 iterations**, paired with an average **runtime of 133.37 seconds** was obtained. It is worth mentioning that the scale of the cube CAD model, which is of the order of $1m$ in each direction, and the dimensions of the mock satellite CAD model, with the span of the solar panels being around $3.5m$, also play a role in this observed performance difference.

As compared to the previous example, in which the inital point cloud was significantly displaced from the CAD's center of geometry, the additonal example shown in Figure 6 features an initial point cloud with a more pronounced rotation misalignment. Similarily, another significant difference between the two scenarios is the complexity, in terms of number of polygons, of the chosen CAD models; the latter omits the addition of the antenna compoment, which, even though appearing to be a rather simple element, drastically increases the count of triangular meshes due to the curved object. This is an important factor, since the resulting shape for the example in Figure 6 contains $N = 709$ vertices and $M = 1,414$ polygons, amounting to a much faster 6DOF pose estimate calculation – **runtime of 50.93 seconds** – for the same level of accuracy as before.

To assess the algorithm's performance against inputs that more closely represent the use of depth-based maps, segmentation techniques were employed in order to sample point clouds only along the triangular polygons visible from the vantage point of a camera lens. An example of this is shown in Figure 7, in which two appropriate 6DOF pose estimates are achieved (same level of accuracy as in previous examples) by means of incomplete point clouds, starting from a random initial position: for Figure 7a), the average iterations needed for convergence were ; for Figure 7b), an average of 9 iterations with a **runtime of 29.89 seconds** was necessary for full registration. Overall, it was observed that for CAD models with dimensions in the order of 10's of meters, source point clouds of $1e^3$ points in size were more than enough to achieve reasonable accuracy levels.

The advantage of testing the proposed algorithm using a sampling technique for generating the initial source point clouds is the availability of ground truth for each scenario. To assess the behaviour of the algorithm in a more representative case, the disparity map shown in Figure 8 is used as an input to the registration algorithm. An identical registration procedure as the ones previously shown is carried out using disparity-based point clouds, such as the one shown in Figure 8, in

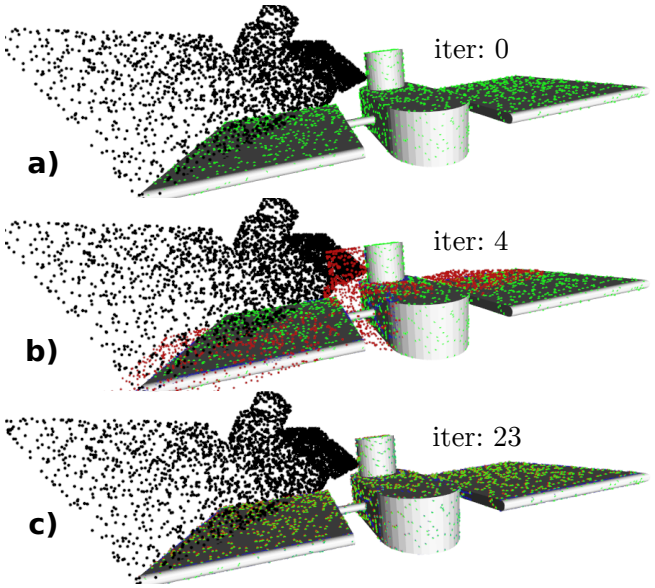


Figure 6. Similar example as the one shown in Figure 5, with the main difference being the omission of the antenna component on the CAD model; the sequence of images shows the registration process, with the black point cloud denoting the scenario's initial source point cloud location (black and red point clouds overlapping in a)). Subfigure b) depicts an intermediate step in which the source point cloud is slowly moving towards the geometric center of the surface; in c), the registered state is shown (red overlapping the green point cloud, which represents the ground truth points).

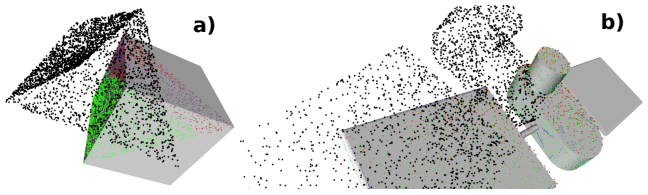


Figure 7. Two registration examples with incomplete point clouds: initial conditions shown in black; ground truth shown in green; final point cloud transformation shown in red.

lieu of sampling-based point clouds, which for the previously shown examples were noiseless. The most important difference with respect to the algorithm's performance between the aforementioned two types of input information is the success rate of the 6DOF pose estimate calculation. For this particular example, convergence was able to be achieve within reasonable time limits only after relaxing the desired accuracy from $1e^{-5}m/\texttt{point}$ to $\eta_d = 5e^{-3}m/\texttt{point}$; an average of **39 iterations** with a **runtime of 180.77 seconds** was obtained. The decrease in achievable accuracy comes from the fact that the point clouds generated from the map $\mathbf{M}$ are not noiseless, thus rendering flat surfaces to be quite irregular (e.g., the ruffled surface shown in Figure 8). This entails that perfect alignment cannot be achieved, since there will always exist some slack between some points in the source point cloud and the surface of the CAD model. For the majority of such cases, the stopping conditions (the Frobenius norm of the difference between the transformation matrices of subsequent iterations falling below $F_{min} = 1e^{-4}$) were triggered before being able to reach the desired accuracy. Further discussion regarding this topic is given in the timing analysis section.

Figure 8. Depth-based information for registration purposes: a) left stereo image of an inspected object; b) disparity map with respect to the left stereo camera; c) disparity map projected as a point cloud.

Timing Analysis

The results shown thus far raise an important topic regarding the distinct trades to be made when employing point clouds and registration methods for real-time operation; it is possible to tradeoff a model's accuracy (e.g., simplifying its shape) for registration runtime, if lower levels of accuracy can be tolerated. Thus, a runtime performance vs point cloud size and CAD model complexity is shown in Figure 9.

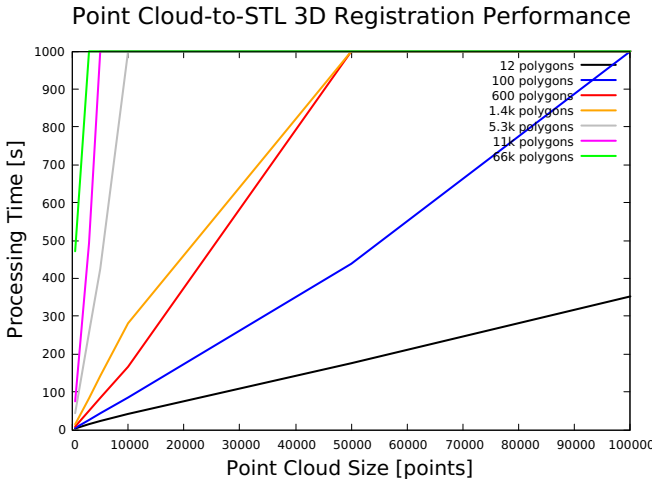


Figure 9. Processing time analysis for distinct CAD model and point cloud sizes. Max processing time was set at $1,000s$.

The distinct color curves represent registration cases employing CAD models with distinct levels of detail; STL files that more coarsely describe an object, such as the blocky ones in which curved segments are discretized into much fewer components, tend to allow for much shorter registration periods (smaller line slope). It can be seen that if very dense point clouds are warranted, the use of complex CAD models becomes prohibitively expensive, with registration time costs quickly ramping up, and vice versa.

Therefore, it is interesting to analyze the complexity of the algorithm's steps with respect to the size of the inputs chosen for the registration. Thus, the separate algorithm steps can be characterized as following: the preprocessing step *i)* is to be performed only once for each employed CAD model, allowing for the ability to carry out such task offline, hence having no impact on the real-time performance of the algorithm; for each time step t taken, a *ii)* reprojection step must be performed, which is a linear in the number of elements inside the depth-based map, that is, of complexity $\mathcal{O}(p{\times}q)$. The calculation of the target point cloud, or step *iii)*, warrants the projection of each point in the point cloud onto the mesh of the CAD model, thus being linear in the number of elements inside the point cloud, $\mathcal{O}(n)$. For each element, k-nearest neighbors need to be computed – $\mathcal{O}(n^{2/3}+k)$ – and afterwards, the interesting polygons need to be retrieved – $\mathcal{O}(M)$. Projecting each point cloud point onto the mesh, and subsequently finding the best one, is linearly dependent on both n and k, $\mathcal{O}(nk)$, hence dominated by the specifications of the input data. Ultimately, the algorithm's time complexity can be parameterized by the number of triangular meshes M, the number of interesting polygons $k \cdot b$, with b acting as a type of branching factor dependent upon the complexity of the CAD model (to how many polygons can a single vertex belong), and the number of points in the source point cloud n. Overall, the current implementation loses the fractional searching time granted by the kd-tree structure over n given the way in which the search for interesting polygons is carried out. To sum it up, the the algorithm's complexity can be approximated as $\mathcal{O}(n(M + nk))$, with a quadratic cost with respect to the size of the source point cloud.

6. Conclusion

The motivation and development of an algorithm for the estimation of a 6DOF relative pose between an inspector satellite, equipped with a 3D depth sensor, and a target object using point clouds and registration methods was presented. Simulation results of several demonstration cases using distinct types of inputs are presented. A coarse time complexity analysis is carried out, and the approach's advantages and disadvatages are outlined.

Future work will focus on improving the algorithm's performance, as well as success rate, in order to enable real-time close proximity operations and to include its output into the spacecraft's estimation pipeline in an online fashion. With the capabilities shown by such an approach, additional research avenues, such as object recognition tasks given a database of known CAD models, are also open for exploration.

Acknowledgments

The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (#NNX16AT66A) as part of the Smoothing-Based Relative

Navigation & Coded Aperture Imaging program.

REFERENCES

[1] D. Sternberg, A. Hilton, D. Miller, B. McCarthy, C. Jewison, D. Roascio, J. James, and A. Saenz-Otero, "Reconfigurable ground and flight testing facility for robotic servicing, capture, and assembly," in *Aerospace Conference, 2016 IEEE*. IEEE, 2016, pp. 1–13.

[2] D. Sternberg, M. Chodas, C. Jewison, M. Jones, and O. De Weck, "Multidisciplinary system design optimization of on orbit satellite assembly architectures," in *2015 IEEE Aerospace Conference*, 2015, pp. 1–14.

[3] A. Flores-Abad, O. Ma, K. Pham, and S. Ulrich, "A review of space robotics technologies for on-orbit servicing," *Progress in Aerospace Sciences*, vol. 68, pp. 1–26, 2014.

[4] J. A. Christian and S. Cryan, "A survey of lidar technology and its use in spacecraft relative navigation," in *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2013, p. 4641.

[5] D. Fourie, B. Tweddle, S. Ulrich, and A. Saenz Otero, "Vision-based relative navigation and control for autonomous spacecraft inspection of an unknown object," in *AIAA Guidance, Navigation, and Control (GNC) Conference*, 2013, p. 4759.

[6] F. Pomerleau, F. Colas, R. Siegwart *et al.*, "A review of point cloud registration algorithms for mobile robotics," *Foundations and Trends® in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[7] J. R. Wertz, D. F. Everett, and J. J. Puschell, *Space mission engineering: the new SMAD*. Microcosm Press, 2011.

[8] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International journal of computer vision*, vol. 13, no. 2, pp. 119–152, 1994.

[9] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke, "Registration with the point cloud library: A modular framework for aligning in 3-d," *IEEE Robotics & Automation Magazine*, vol. 22, no. 4, pp. 110–124, 2015.

[10] B. Bellekens, V. Spruyt, R. Berkvens, and M. Weyn, "A survey of rigid 3d pointcloud registration algorithms," in *AMBIENT 2014: the Fourth International Conference on Ambient Computing, Applications, Services and Technologies, August 24-28, 2014, Rome, Italy*, 2014, pp. 8–13.

[11] A. Censi, "An icp variant using a point-to-line metric," in *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 2008, pp. 19–25.

[12] P. Yan and K. W. Bowyer, "A fast algorithm for icp-based 3d shape biometrics," *Computer Vision and Image Understanding*, vol. 107, no. 3, pp. 195–202, 2007.

[13] F. Aghili, M. Kuryllo, G. Okouneva, and C. English, "Fault-tolerant position/attitude estimation of free-floating space objects using a laser range sensor," *IEEE Sensors Journal*, vol. 11, no. 1, pp. 176–185, 2011.

[14] F. Aghili and C.-Y. Su, "Robust relative navigation by integration of icp and adaptive kalman filter using laser scanner and imu," *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 4, pp. 2015–2026, 2016.

[15] W. Li and P. Song, "A modified icp algorithm based on dynamic adjustment factor for registration of point cloud and cad model," *Pattern Recognition Letters*, vol. 65, pp. 88–94, 2015.

[16] Y. Ito and K. Nakahashi, "Direct surface triangulation using stereolithography data," *AIAA journal*, vol. 40, no. 3, pp. 490–496, 2002.

[17] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 80–91, 2012.

[18] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[19] Blender Online Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Blender Institute, Amsterdam, 2018. [Online]. Available: http://www.blender.org

BIOGRAPHY

Antonio Terán Espinoza *is a visiting student researcher at the Jet Propulsion Laboratory, and doctoral candidate at the MIT Space Systems Laboratory. He received his B.S. from the National Autonomous University of Mexico (UNAM), and his S.M. degree from the Massachusetts Institute of Technology as a Fulbright student. He is currently an integral member of the SPHERES group, and is working towards his Ph.D. in Aeronautics and Astronautics with a focus in autonomous navigation, mapping, and perception.*

Timothy P. Setterfield *is a Guidance and Control Engineer at the Jet Propulsion Laboratory in Pasadena, CA. He holds a BScE in Mechanical Engineering from Queens University, Kingston, ON, Canada, a MASc in Mechanical and Aerospace Engineering from Carleton University, Ottawa, ON, Canada, and a PhD in Aeronautics and Astronautics from the Massachusetts Institute of Technology. He has previously worked at Nanometrics Seismological Instruments developing a miniature broadband seismometer, at Carleton University developing a planetary micro-rover prototype, at the European Space Agency coordinating graduate student research in micro and hypergravity, and at MIT researching vision-based navigation using the SPHERES-VERTIGO platform.*